

REMARKS

This paper is responsive to the Office Action dated May 24, 2008. All rejections and objections of the Examiner are respectfully traversed. Reconsideration and further examination are respectfully requested.

The amendments to the claims are intended to clarify the claim language and more precisely set forth the present invention.

No new matter has been added.

At paragraph 3 of the Office Action, the Specification was objected to for informality. Amendments to the Specification herein are respectfully believed to meet all requirements in this regard.

At paragraphs 4-5 of the Office Action, the Examiner rejected claims 1, 2, 5-11, 13-15 and 20-23 for indefiniteness under 35 U.S.C. 112, second paragraph. Amendments to the claims herein are respectfully believed to meet all requirements in this regard.

At paragraphs 6-21 of the Office Action, the Examiner rejected claims 1, 2, 5-11 and 13-15 for obviousness under 35 U.S.C. 103, citing the combination of U.S. patent 6,691,067 of Ding et al. ("Ding") and U.S. patent application publication 2003/0056200 ("Li"). Applicant respectfully traverses these rejections.

As noted previously, Ding discloses an enterprise management system that includes statistical recreation of system resource usage for monitoring, prediction, and performance workload characterization. In Ding, an agent computer system includes agent software for collecting data relating to one or more metrics, i.e., measurements of system resources. Metric data is continually collected by the agent software of Ding over the course of a measurement

interval, regularly placed into a registry of metrics, and then periodically sampled from the registry indirectly.

Li discloses a monitoring system for component-based software system that includes initiating an invocation of a second software component from within an execution of a first software component. Stub start log data is recorded in an instrumented stub by Li invocation of the second software component. A stub end log data is recorded in the instrumented stub of Li after a response is received from the invocation of the second software component. The stub start log data and the stub end log data of Li gather runtime information about execution of the second software component within the component-based software system. The monitoring performed by Li is capable of gathering log data across a plurality of threads, across a plurality of processes, and across a plurality of processors, and the log data may be retrieved and analyzed to produce timing latency information, shared resource usage information, application semantics information, and causality relationship information.

The software components of Li are described as objects in paragraph 0027. The objects of Li are described in paragraph 0031 as computational groupings of operations and data into modular units. Li teaches further that the objects of Li are defined by the interfaces they present to others, their behavior when operations on their interfaces are invoked, and their state. In paragraph 0032 Li teaches that a thread is an independent execution sequence of program code inside a process, that threads are execution entities through which the function invocations are carried, and that one or more threads are caused by and related to a function call. The software components of Li are composed of sets of functions, each of which has a defined function interface. Specific log data is created for each function call invocation in Li. See paragraph 0040.

Li describes determining function latency in paragraphs 0081-0084, and shared resource usage monitoring beginning in paragraph 0085. CPU usage by a distributed function call is described beginning at paragraph 0094. Li describes viewing CPU consumption as a propagation activity, by following defined causality relationships, including both function caller/callee relationships and thread parent/child relationships, in order to determine all of the CPU consumption from the involved child function calls, as well as the dynamically spawned threads in the middle of such function invocations. In paragraph 0116, Li teaches that probes may be inserted into the thread library to determine the lifetime span of a thread and determine the resource usage and parent-child relationships.

Nowhere in the combination of Ding and Li is there disclosed or suggested a method for monitoring system processor resources utilized by a software agent operating in a computer system, wherein said agent comprises an executable sequence of instructions, said method comprising the steps of:

identifying said agent, wherein said identifying is performed by a CPU resource tracking process determining that said agent is running;

associating, by said CPU resource tracking process responsive to said identifying of said agent, an agent identifier with said agent, wherein said agent is one of a plurality of software agents operating in said computer system, and wherein said agent identifier uniquely identifies said agent within said plurality of software agents operating in said computer system;

initiating, responsive to said identifying of said agent, an agent lifetime timer for measuring an operating interval of said agent during which said agent is running in said computer system;

determining said operating interval using said agent lifetime timer by identifying a start time at which said CPU resource tracking process determined that said agent is running and a completion time at which said CPU resource tracking process determines that said agent has expired, and computing said operating interval as the difference between said starting time and said completion time;

calculating an amount of said system processor resources utilized by said agent during said operating interval at least in part by detecting a plurality of threads created by said agent during said operating interval, wherein said calculating said amount of

said system processor resources utilized by said agent during said operational interval further includes calculating CPU usage for each of said plurality of threads and adding said calculated CPU usage for all of said plurality of threads together to determine at least a portion of said amount of system processor resources utilized by said agent during said operating interval, wherein said calculating said CPU usage for each of said plurality of threads is performed responsive to each respective one of said plurality of threads expiring, and wherein said adding said calculated CPU usage for all of said plurality of threads together is performed by adding said calculated CPU utilization for each respective one of said plurality of threads responsive to each respective one of said plurality of threads expiring, wherein each of said plurality of threads is a path of execution such that multiple of said plurality of threads can be executed simultaneously; and

storing said operating interval, said amount of system processor resources utilized by said agent during said operating interval and said agent identifier in a computer-readable memory. (emphasis added)

as in the present independent claim 1. Combining Ding with Li results in a system that uses a software agent to monitor an agent computer system, as in Ding, and that uses probes inserted into function calls and thread libraries to determine function and thread latency as well as CPU consumption of a function, as in Li. Nothing in the combination of Ding and Li teaches or suggests initiating, responsive to identifying an agent, an agent lifetime timer for measuring an operating interval of the agent during which said agent is running, determining the operating interval using the agent lifetime timer by identifying a start time at which a CPU resource tracking process determined that the agent is running and a completion time at which the CPU resource tracking process determines that said agent has expired, and computing the operating interval as the difference between the starting time and said completion time, calculating an amount of the system processor resources utilized by the agent during the operating interval at least in part by detecting a plurality of threads created by the agent during the operating interval, wherein said calculating said amount of said system processor resources utilized by said agent during said operational interval further includes calculating CPU usage for each of the plurality of threads and adding the calculated CPU usage for all of the plurality of threads together to

determine at least a portion of the amount of system processor resources utilized by the agent during the operating interval, wherein the calculating of the CPU usage for each of the plurality of threads is performed responsive to each respective one of the plurality of threads expiring, and wherein said adding the calculated CPU usage for all of the plurality of threads together is performed by adding the calculated CPU utilization for each respective one of said plurality of threads responsive to each respective one of said plurality of threads expiring, as in the present independent claim 1. In contrast, measuring is not performed with regard to the CPU utilization of the software agent in Ding, and the monitoring of Li is with respect to function calls.

Accordingly, the combination of Ding and Li does not disclose or suggest all the features of the present independent claims 1. The combination of Ding and Li therefore does not support a *prima facie* case of obviousness under 35 U.S.C. 103 with regard to independent claim 1. As to dependent claims 2, 5-11 and 13-15, they each depend from independent claim 1, and are believed to be patentable over Ding and Li for at least the same reasons.

In paragraphs 22-28 of the Office Action, the Examiner rejected claims 20-23 for obviousness under 35 U.S.C. 103, based on the combination Ding, Li and U.S. patent number 6,330,588 ("Freeman"). Applicant respectfully traverses this rejection.

Freeman discloses a system for verification of software agents and their activities in a distributed computing environment including an origin resource, a destination resource and a trusted resource. The origin resource of Freeman is associated with a software agent. The destination resource of Freeman is expected to advance the agent in the performance of an entrusted task, and the trusted resource is associated with the software agent in that the trusted resource functions to provide verification of the software agent and its activities. The trusted

resource of Freeman supports one or more selected operations such as receiving/forwarding of software agents and other operations.

Like the combination of Ding and Li, the combination of Ding, Li and Freeman does not disclose or suggest any initiating, responsive to identifying an agent, an agent lifetime timer for measuring an operating interval of the agent during which said agent is running, determining the operating interval using the agent lifetime timer by identifying a start time at which a CPU resource tracking process determined that the agent is running and a completion time at which the CPU resource tracking process determines that said agent has expired, and computing the operating interval as the difference between the starting time and said completion time, calculating an amount of the system processor resources utilized by the agent during the operating interval at least in part by detecting a plurality of threads created by the agent during the operating interval, wherein said calculating said amount of said system processor resources utilized by said agent during said operational interval further includes calculating CPU usage for each of the plurality of threads and adding the calculated CPU usage for all of the plurality of threads together to determine at least a portion of the amount of system processor resources utilized by the agent during the operating interval, wherein the calculating of the CPU usage for each of the plurality of threads is performed responsive to each respective one of the plurality of threads expiring, and wherein said adding the calculated CPU usage for all of the plurality of threads together is performed by adding the calculated CPU utilization for each respective one of said plurality of threads responsive to each respective one of said plurality of threads expiring, as in the present independent claim 1. Applicant accordingly respectfully urges that the combination of Ding, Li and Freeman does not disclose or suggest all the features of the present independent claim 1. The combination of Ding, Li and Freeman therefore does not support a

prima facie case of obviousness under 35 U.S.C. 103 with regard to claim 1. As to claims 20-23, they each depend from claim 1, and are respectfully believed to be patentable over the combination of Ding, Li and Freeman for at least the same reasons.

Applicant has amended the claims herein, but Applicant is not conceding in this application that the unamended claims are not patentable over the art cited by the Examiner, as the present claim amendments are only for facilitating expeditious prosecution of allowable subject matter. Applicant respectfully reserves the right to pursue the unamended claims in one or more continuations and/or divisional patent applications.

Applicant has made a diligent effort to place the claims in condition for allowance. However, should there remain unresolved issues that require adverse action, it is respectfully requested that the Examiner telephone Applicant's Attorney at the number listed below so that such issues may be resolved as expeditiously as possible.

For these reasons, and in view of the above amendments, this application is now considered to be in condition for allowance and such action is earnestly solicited.

Respectfully Submitted,

May 26, 2009

Date

/David Dagg/

David Dagg, Reg. No. 37,809
Attorney/Agent for Applicant(s)
44 Chapin Road
Newton, MA 02459
(617) 630-1131

Docket No. 260-078